REMARKS

Claims 1-69 are all the claims presently pending in the application. Claims 9, 26, 36, 39-40, 43-50, 53, 55, 61, 66 and 69 have been amended to more clearly define the invention.

It is noted that the claim amendments herein are made only for more particularly pointing out the invention, and not for distinguishing the invention over the prior art, narrowing the claims, or for any statutory requirements of patentability.

Further, it is noted that, notwithstanding any claim amendments made herein, Applicant's intent is to encompass equivalents of all claim elements, even if amended herein or later during prosecution.

Claims 9, 26, 36, 40, 43-48, 50 and 55 stand rejected under 35 U.S.C. § 112, second paragraph as indefinite.

Claim 1 stands rejected under the judicially created doctrine of double patenting over claim 1 of co-pending Application No. 09/569,308 (hereinafter "the '308 application"), claim 60 stands rejected under the judicially created doctrine of double patenting over claim 31 of the '308 application, claims 1 and 67 stand rejected under the judicially created doctrine of double patenting over claim 1 of U.S. Patent No. 6,101,524 (hereinafter, "the '524 patent").

Claims 1-33, 35-48, 57-62, and 64-69 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over "Deterministric Replay of Java Multithreaded Applications", by John-Deok Choi, et al. (hereinafter "the Choi/Srinivasan Article") in view of "TCP/IP Illustrated, Volume 1, The Protocols", by W. Richard Stevens (1994) (hereinafter, "the Stevens book"). Claims 34 and 49-56 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the Choi/Srinivasan Article in view of the Stevens book, and further in view of "The JAVA Series from the Source", by Patrick Chan (hereinafter, "the Chan book").

These rejections are respectfully traversed in view of the following discussion.

I. THE CLAIMED INVENTION

The claimed invention (e.g., as recited in claim 1) is directed to a method for recording and replaying execution of distributed programs on a computer system in a distributed environment. The method includes identifying an execution order of critical events of a program, generating groups of critical events of the program, wherein for each group, critical events belonging to the group belong to a common execution thread, and generating, for each execution thread, a logical thread schedule that identifies a sequence of the groups so as to allow deterministically replaying a non-deterministic arrival of stream socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages.

Another exemplary aspect of the claimed invention (e.g., as recited in claim 61) is directed to a method for supporting execution replay with respect to datagram socket Application Programming Interface (API). The method includes identifying an execution order of critical events of a program, generating groups of critical events of said program, wherein for each group, critical events belonging to said group belong to a common execution thread, determining out-of-order delivery of packets, and determining a non-deterministic number of packets delivered during different executions of the same program, for supporting an execution replay with respect to said datagram socket Application Programming Interface (API).

Conventional methods do not allow for efficient recording and replaying execution of distributed programs on a computer system in a distributed environment (Application at page 6, lines 2-7).

The claimed invention, on the other hand, includes in one aspect, a method which includes generating, for each execution thread, a logical thread schedule that identifies a sequence of the groups so as to allow deterministically replaying a non-deterministic arrival of stream socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages. Another aspect of the claimed invention includes a method which

includes determining a non-deterministic number of packets delivered during different executions of the same program, for supporting an execution replay with respect to said datagram socket Application Programming Interface (API).

These features allow the claimed invention to provide for efficient recording and replaying execution of distributed programs on a computer system in a distributed environment Application at page 6, lines 9-16).

II. THE 35 U.S.C. §112, SECOND PARAGRAPH REJECTION

The Examiner alleges that claims 9, 26, 36, 40, 43-48, 50 and 55 are indefinite. Applicant submits however that these claims are clearly defined and not indefinite.

Further, Applicant notes that these claims have been amended to address the Examiner's concerns.

Reconsideration and withdrawal of these rejections are respectfully requested.

III. THE DOUBLE PATENTING REJECTIONS

The Examiner alleges that claim 1 is made obvious by claim 1 of the '308 application, that claim 60 is made obvious by claim 31 of the '308 application, and that claims 1 and 67 are made obvious by claim 1 of the '524 patent. Applicant submits, however, that the neither claims 1 and 31 of the '308 application, nor claim 1 of the '524 patent teaches or suggests each and every element of the present Application.

First, Applicant would point out to the Examiner that he cannot rely on the disclosure of the specification in the '524 patent or the '308 application, to support his allegation of double patenting. Instead, the Examiner is limited only to the disclosure of the claims of the '524 patent and the '308 application. Clearly, the claims of the '524 patent and the '308 application fall very short of teaching or suggesting the claimed invention of the present Application.

Claim 1 of the '524 patent discloses a program storage device for performing a method of recording a representation of run-time behavior of a program ('524 patent at col. 12, lines 25-50).

Claims 1 and 31 of the '308 application disclose a method of deterministically replaying

an observable run-time behavior of distributed multi-threaded programs on multiprocessors in a shared-memory multiprocessor environment.

However, neither claim 1 of the '524 patent, nor claims 1 and 31 of the '308 application teaches or suggests a method for recording and replaying execution of distributed programs on a computer system in a distributed environment, which includes "*generating, for each execution thread, a logical thread schedule that identifies a sequence of said groups* ***so as to allow deterministically replaying a non-deterministic arrival of stream socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages***" as recited, for example, in claim 1 and similarly recited in claim 67.

Clearly, these features are not taught or suggested by claim 1 of the '524 patent, or claims 1 and 31 of the '308 application. Indeed, these references are completely unrelated to the claimed invention.

Specifically, with respect to claim 1 of the '524 patent, Applicant would point out that the present Application expressly states that the claimed invention is an improvement over the method disclosed in the '524 patent (Application at page 6, lines 12-16). Further, the claims in the '524 patent do not each the approach to replay network I/O operations. Indeed, the '524 patent only works for one single JVM and replay of multithreaded applications that do not have any network operations. With network operations, multiple JVM's are involved hence requiring the identification of which JVM generated a network operation and capture of these network operations at each participating JVM in the replay mechanism.

Unlike the claims of the '524 patent and the '308 application, the claimed invention provides details of how network I/O (e.g., socket and datagram reads, writes, listen, bind, etc.,) in Java can be replayed. For example, in one exemplary embodiment, unique VM-id's may be used to keep track of the sender and receiver of the network events. The VM-id is not something that can be inferred from the claims of the '524 paper or the '308 application.

Therefore, Applicant respectfully submits that there are elements of the claimed invention that are not taught or suggested by the claims of the '524 patent or the '308 application.

Therefore, the Examiner is respectfully requested to withdraw this rejection.

IV. THE ALLEGED PRIOR ART REFERENCES

A. The Choi/Srinivasan Article and the Stevens Book

The Examiner alleges that the Choi/Srinivasan Article and the Stevens Book would have been combined to form the claimed invention of claims 1-33, 35-48, 57-62 and 64-69. Applicant submits, however, that these references would not have been combined and even if combined, the combination would not teach or suggest each and every element of the claimed invention.

The Choi/Srinivasan Article discloses a record/replay tool for Java applications, called DejaVu that provides a deterministic replay of a non-deterministic execution. DejaVu is independent of the underlying thread scheduler and runs on a uniprocessor system (Choi/Srinivasan Article at page 12, left column).

The Stevens book discloses two application programming interfaces (APIs) for applications using the TCP/IP protocols, sockets and transport layer interface (TPI) (Stevens at page 17).

However, Applicant submits that these references would not have been combined as alleged by the Examiner. Indeed, these references are directed to different problems and solutions.

Specifically, the Choi/Srinivasan Article is directed to a record/replay tool for Java applications, whereas Stevens is merely directed to a TCP/IP protocols. Therefore, these references are completely unrelated, and no person of ordinary skill in the art would have considered combining these disparate references, absent impermissible hindsight.

Further, Applicant submits that the Examiner can point to no motivation or suggestion in the references to urge the combination as alleged by the Examiner. Indeed, contrary to the Examiner's allegations, neither of these references teach or suggest their combination. Therefore, Applicant respectfully submits that one of ordinary skill in the art would not have been so motivated to combine the references as alleged by the Examiner. Therefore, the Examiner has failed to make a prima facie case of obviousness.

Moreover, neither the Choi/Srinivasan Article, nor the Stevens book, nor any combination thereof teaches or suggests a method for recording and replaying execution of distributed programs on a computer system in a distributed environment, which includes "*generating, for each execution thread, a logical thread schedule that identifies a sequence of said groups* ***so as to allow deterministically replaying a non-deterministic arrival of stream socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages***" as recited, for example, in claim 1 and similarly recited in claims 60, 64-65 and 67-68, or "*determining a non-deterministic number of packets delivered during different executions of the same program,* ***for supporting an execution replay with respect to said datagram socket Application Programming Interface (API)***", as recited in claims 61, 66 and 69.

Clearly, these features are not taught or suggested by the Choi/Srinivasan Article. Indeed, the Choi/Srinivasan Article is completely unrelated to the claimed invention. In fact, the Examiner expressly concedes that the Choi/Srinivasan Article does not teach or suggest this feature.

Similarly, the Stevens book does not teach or suggest this feature. Indeed, as noted above, the Stevens book merely discloses two application programming interfaces (APIs) for applications using the TCP/IP protocols, sockets and transport layer interface (TPI). Nowhere does the Stevens book teach or suggest ***deterministically replaying a non-deterministic arrival of stream socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages***, or *determining a non-deterministic number of packets delivered during different executions of the same program,* ***for supporting an execution replay with respect to said datagram socket Application Programming Interface (API)***", as in the exemplary aspects of the claimed invention.

Therefore, Applicant submits that these references would not have been combined, and even if combined the combination would not teach or suggest each and every element of the

claimed invention. Therefore, the Examiner is respectfully requested to withdraw this rejection.

B. The Chan Book

The Examiner alleges that the Choi/Srinivasan Article and the Stevens Book would have been combined with the Chan book to form the claimed invention of claims 34 and 49-56. Applicant submits, however, that these references would not have been combined and even if combined, the combination would not teach or suggest each and every element of the claimed invention.

The Chan book discloses JAVA packages, classes and methods.

However, Applicant submits that these references would not have been combined as alleged by the Examiner. Indeed, these references are directed to different problems and solutions.

Specifically, the Chan book merely describes JAVA packages, classes and methods, which is unrelated to the record/replay tool of the Choi/Srinivasan Article or the TCP/IP protocols of the Stevens book. Therefore, these references are completely unrelated, and no person of ordinary skill in the art would have considered combining these disparate references, absent impermissible hindsight.

Further, Applicant submits that the Examiner can point to no motivation or suggestion in the references to urge the combination as alleged by the Examiner. Indeed, contrary to the Examiner's allegations, neither of these references teach or suggest their combination. Therefore, Applicant respectfully submits that one of ordinary skill in the art would not have been so motivated to combine the references as alleged by the Examiner. Therefore, the Examiner has failed to make a prima facie case of obviousness.

Moreover, neither the Chan book, nor the Choi/Srinivasan Article, nor the Stevens book, nor any combination thereof teaches or suggests a method for recording and replaying execution of distributed programs on a computer system in a distributed environment, which includes "*generating, for each execution thread, a logical thread schedule that identifies a sequence of said groups* ***so as to allow deterministically replaying a non-deterministic arrival of stream***

socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages" as recited, for example, in claim 1.

Clearly, these features are not taught or suggested by the Chan book. Indeed, the Chan book is completely unrelated to the claimed invention.

As noted above, the Chan book merely discloses JAVA packages, classes and methods. Nowhere does Chan teach or suggest a method for recording and replaying execution of distributed programs on a computer system in a distributed environment, let alone such a method which includes generating, for each execution thread, a logical thread schedule that identifies a sequence of said groups **so as to allow deterministically replaying a non-deterministic arrival of stream socket connection requests, a non-deterministic number of bytes received during message reads, a non-deterministic binding of stream sockets to local ports, and a non-deterministic arrival of datagram messages.**

Moreover, Applicant submits that multiprocessor applications do not imply that they always have network I/O operations in them. Thus, it cannot be easily deduced, by anyone familiar with Steven's teachings, the techniques to implement replay of network I/O. Similarly, it cannot be easily deduced, by anyone familiar with Chan's teachings, the techniques to implement replay of network I/O.

With network operations, multiple JVM's are involved hence requiring the identification of which JVM generated a network operation and capture of these network operations at each participating JVM in the replay mechanism.

The claimed invention provides details of how network I/O (e.g., socket and datagram reads, writes, listen, bind, etc.,) in Java can be replayed. For example, claim 36 talks about unique VM-id's to keep track of the sender and receiver of the network events. The VM-id is not something that can be inferred easily from the Choi/Srinivasan Article.

Further, the teachings of Stevens and Chan cannot be used in combination with the Choi/Srinivasan Article to deduce the VM-id idea in a straightforward manner. Likewise, other specific network I/O replay independent claims include 37 (that elucidates network id), claim 38

(connection id), claim 39 (threadNum), claim 40-41 (how the connection id is sent over the network as well), claim 42 (when the connection id is sent), claim 43 (network log file creation and population in record phase), claim 44 (client event id and server socket log), claim 45 (identifying connection id in the replay phase) and claims 46-69 that further claim the replay mechanism of socket and datagram network I/O replay.

Therefore, Applicant submits that these references would not have been combined, and even if combined the combination would not teach or suggest each and every element of the claimed invention. Therefore, the Examiner is respectfully requested to withdraw this rejection.

V. FORMAL MATTERS AND CONCLUSION

The Examiner objects to the drawings. Applicant notes that a Submission of Replacement Drawing Sheet is submitted herewith for Figure 11, to insert the designation "1100" which was inadvertently omitted. Applicant further notes that the specification has been amended to address the remainder of the Examiner's drawing objections.

The Examiner objects to the specification. Applicant notes that the specification has been amended to address the Examiner's objections thereto.

Applicant notes that claim 53 has been amended to address the Examiner's objections thereto.

In view of the foregoing, Applicant submits that claims 1-69, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

Serial No.: 09/520,008 30
Docket No. YO999-502
YOR.149

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Phillip E. Miller
Registration No. 46,060

Date: 1/30/04

McGinn & Gibb, PLLC
Intellectual Property Law
8321 Old Courthouse Road, Suite 200
Vienna, Virginia 22182-3817
(703) 761-4100
Customer No. 21254

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that the foregoing Amendment was filed by facsimile with the United States Patent and Trademark Office, Examiner Mary Steelman, Group Art Unit # 2122 at fax number (703) 872-9306 this 30th day of January, 2004.

Phillip E. Miller
Reg. No. 46,060